Algoritma Runut-balik (Backtracking)

Bahasan

1. Pendahuluan
2. Penyelesaian dengan *Backtracking*
3. Algoritma Runut-balik
4. Properti Umum Metode Runut-balik
5. Pengorganisasian Solusi
6. Prinsip Solusi dengan Metode Runut-balik
7. Skema Algoritma Runut-Balik (versi rekursif)
8. Pewarnaan Graf (*Graph Colouring)*

1. Pendahuluan

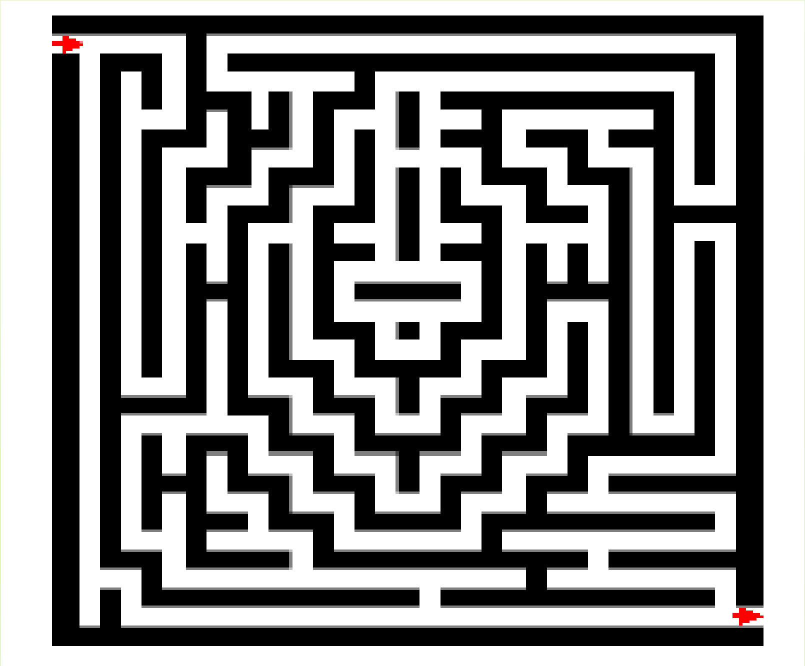
- ***Backtracking* dapat dipandang sebagai salah satu dari dua hal berikut:**

1. **Sebagai sebuah fase di dalam algoritma traversal DFS**
2. **Sebagai sebuah metode pemecahan masalah yang mangkus, terstruktur, dan sistematis**

- **Runut-balik banyak diterapkan untuk program *games* :**
  - **permainan *tic-tac-toe*,**
  - **menemukan jalan keluar dalam sebuah labirin,**
  - **Catur, *crossword puzzle*, *sudoku*, dan masalah-masalah pada bidang kecerdasan buatan (*artificial intelligence*).**

1. Pendahuluan

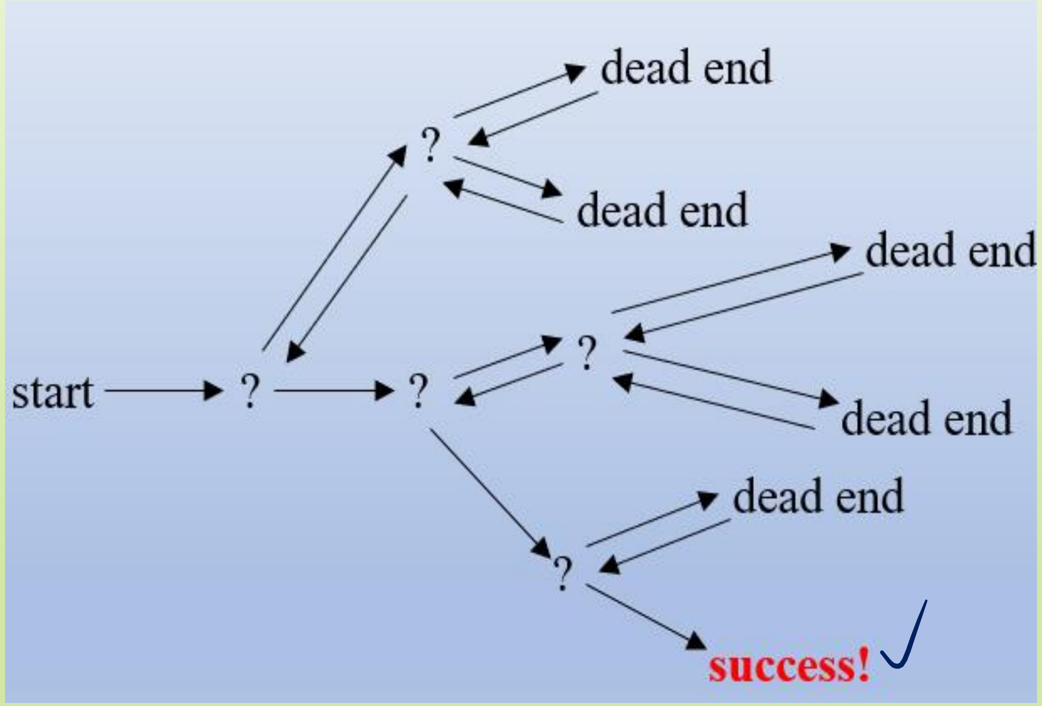
- Contoh (*Maze problem*): diberikan sebuah labirin (*maze*), temukan lintasan dari titik awal sampai titik akhir



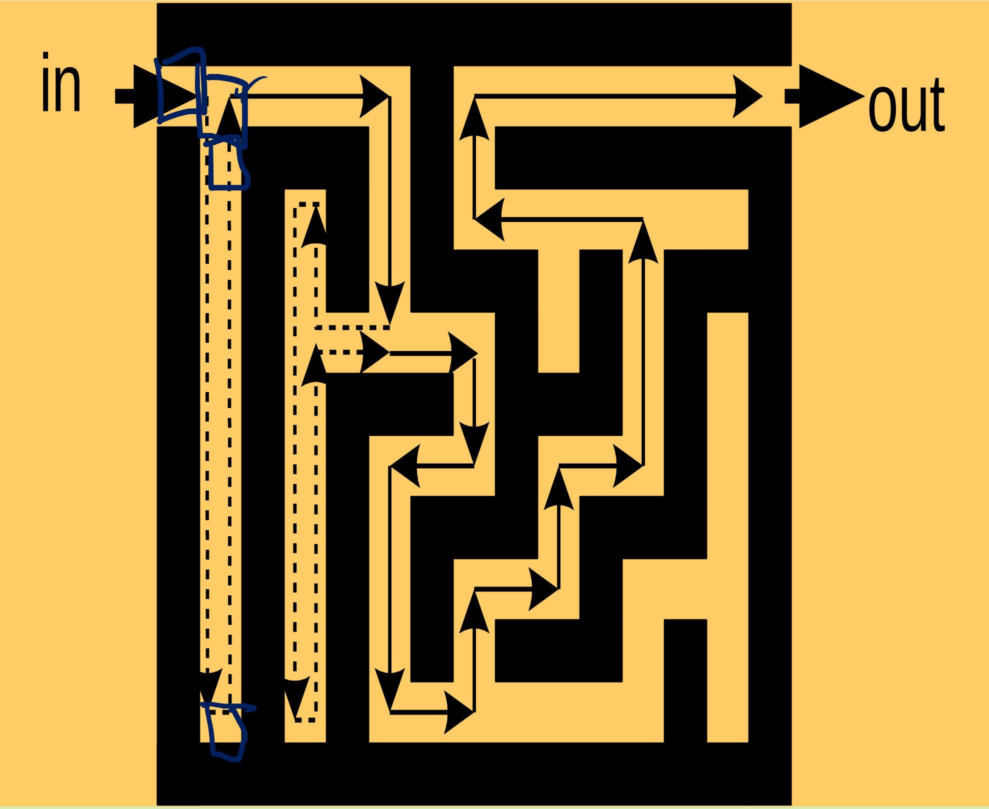
- Pada tiap perpotongan, anda harus memutuskan satu diantara tiga pilihan:
  - Maju terus
  - Belok kiri
  - Belok kanan
- Anda tidak punya cukup informasi untuk memilih pilihan yang benar (yang mengarah ke titik akhir)
- Tiap pilihan mengarah ke sekumpulan pilihan lain
- Satu atau lebih sekuens pilihan mengarah ke solusi.
- *Backtracking* (runut-balik) dapat digunakan untuk persoalan seperti ini

1. Pendahuluan

- **Animasi backtrack**



*) Sumber: *www.cis.upenn.edu/.../35-**backtracking**.ppt*



2. Penyelesaian dengan *Backtracking*

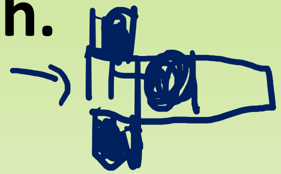
- **Bagi lintasan menjadi sederetan langkah.**
- **Sebuah langkah terdiri dari pergerakan satu unit sel pada arah tertentu.**
- **Arah yang mungkin: lurus (*straight*), kiri (*left*), ke kanan (*right*).**

- **Algoritma umum RB**

```
while belum sampai pada tujuan do
   if terdapat arah yang benar sedemikian sehingga kita belum pernah
        berpindah ke sel pada arah tersebut
   then
        pindah satu langkah ke arah tersebut
   else
        backtrack langkah sampai terdapat arah seperti yang disebutkan
        di atas
   endif
endwhile
```

2. Penyelesaian dengan *Backtracking*

- **Bagaimana mengetahui langkah yang mana yang perlu dijejaki kembali?**
- **Ada dua solusi untuk masalah ini:**
  1. **Simpan semua langkah yang pernah dilakukan, atau**
  2. **gunakan rekursi (yang secara implisit menyimpan semua langkah).**
- **Rekursi adalah solusi yang lebih mudah.**

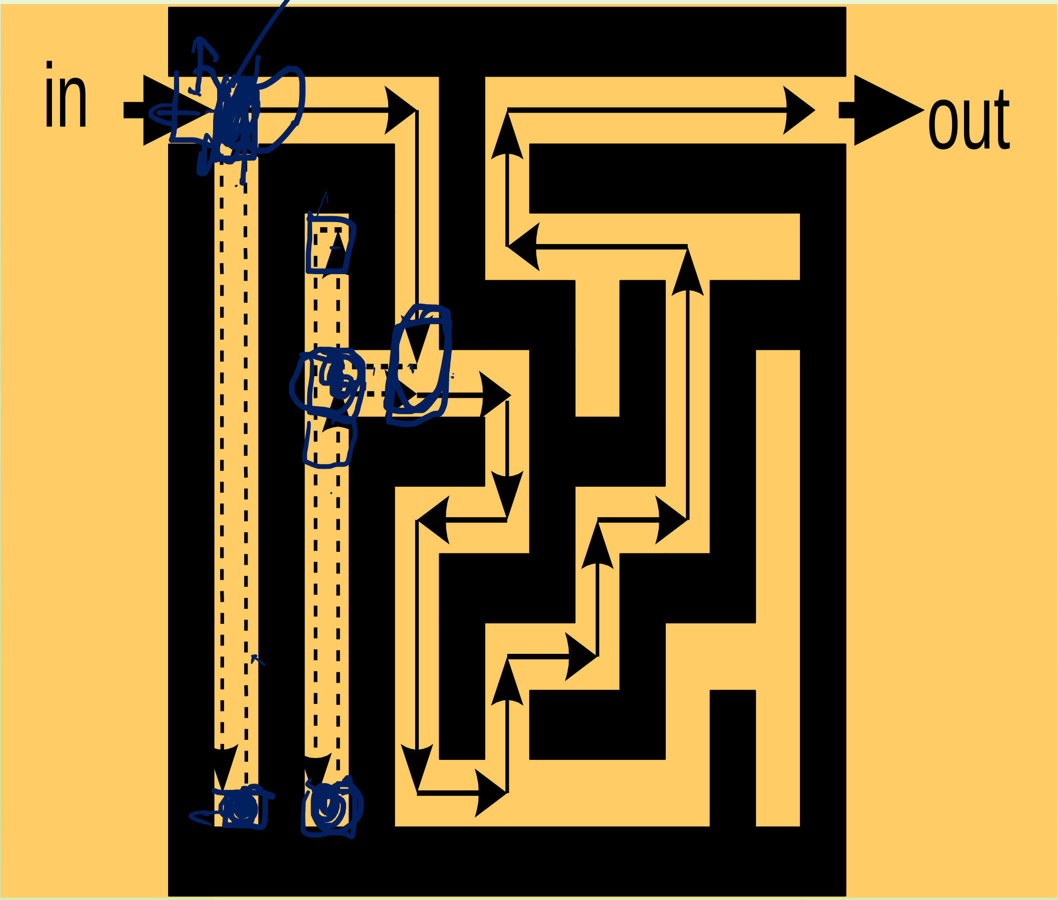


```
function SolveMaze(input M : labirin)→boolean
Deklarasi
   arah : integer   { up = 1, down, 2, left = 3, right = 4 }
Algoritma:
   if pilihan arah merupakan solusi then
     return true
   else
                                         up, dw, lf, rt
     for tiap arah gerakan (lurus, kiri, kanan) do
        move(M, arah)   { pindah satu langkah (satu sel)
                          sesuai arah  tersebut }
        if SolveMaze(M) then
          return true
        else
          unmove(M, arah)   { backtrack }
        endif
     endfor
     return false     { semua arah sudah dicoba, tetapi
                        tetap buntu,  maka
                        kesimpulannya: bukan solusi }
   endif
```

2. Penyelesaian dengan *Backtracking*

up x dw ✓ left x
right

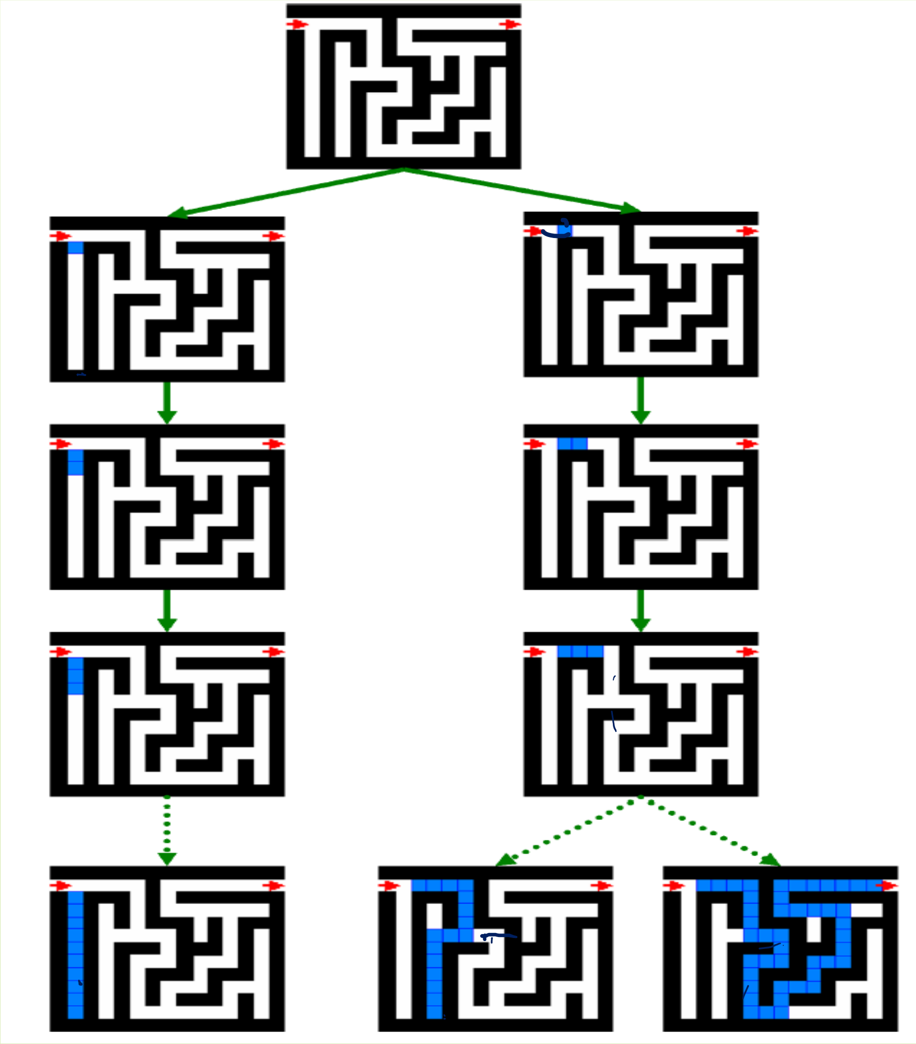
- **Contoh runut-balik pada sebuah labirin.**
- **Runut-balik diperlihatkan dengan garis putus-putus.**



up, down, left, right

2. Penyelesaian dengan *Backtracking*

- **Jika kita menggambarkan sekuens pilihan yang kita lakukan, maka diagram berbentuk seperti pohon.**
- **Simpul daun merupakan:**
  1. **Titik *backtrack*, atau**
  2. **Simpul *goal***
- **Pada titik *backtrack*, simpul tersebut menjadi mati (tidak bisa diekspansi lagi)**
- **Aturan pembentukan simpul: DFS**



3. Algoritma Runut-balik

- **Algoritma runut-balik merupakan perbaikan dari *exhaustive search*.**
- **Pada *exhaustive search*, semua kemungkinan solusi dieksplorasi satu per satu.**
- **Pada *backtracking*, hanya pilihan yang mengarah ke solusi yang dieksplorasi, pilihan yang tidak mengarah ke solusi tidak dipertimbangkan lagi**

  → Memangkas (*pruning*) simpul-simpul yang tidak mengarah ke solusi.

- **Algoritma runut-balik pertama kali diperkenalkan oleh D. H. Lehmer pada tahun 1950.**
- **R.J Walker, Golomb, dan Baumert menyajikan uraian umum tentang algoritma runut-balik.**

4. Properti Umum Metode Runut-balik

1. **Solusi persoalan.**
   - **Solusi dinyatakan sebagai vektor dengan *n-tuple*: $X = (x_1, x_2, \ldots, x_n)$, $x_i \in S_i$.**
   - **Mungkin saja $S_1 = S_2 = \ldots = S_n$.** $S_1 = \{a, b, \ldots$
   - **Contoh: $S_i = \{0, 1\}$, $x_i = 0$ atau 1** $S_i = \{1, 2, \ldots 9$
2. **Fungsi pembangkit nilai x_k**

   Dinyatakan sebagai predikat:

   $$T(k)$$

   $T(k)$ membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.
3. **Fungsi pembatas**
   - **Dinyatakan sebagai predikat**

     $$B(x_1, x_2, \ldots, x_k)$$
   - **B bernilai *true* jika $(x_1, x_2, \ldots, x_k)$ mengarah ke solusi.**
   - **Jika *true*, maka pembangkitan nilai untuk x_{k+1} dilanjutkan, tetapi jika *false*, maka $(x_1, x_2, \ldots, x_k)$ dibuang.**

5. Pengorganisasian Solusi

$2^3 = 8$

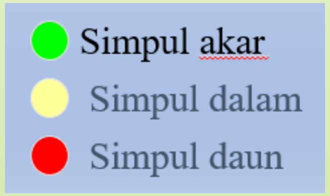
- **Semua kemungkinan solusi dari persoalan disebut ruang solusi (*solution space*).**
- **Tinjau *Knapsack* 0/1 untuk $n = 3$.**
- **Solusi persoalan dinyatakan sebagai (x_1, x_2, x_3) dengan $x_i \in \{0,1\}$.**
- **Ruang solusinya adalah:**

  {(0, 0, 0),(0, 1, 0), (0, 0, 1), (1, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 1), (1, 1, 1)}

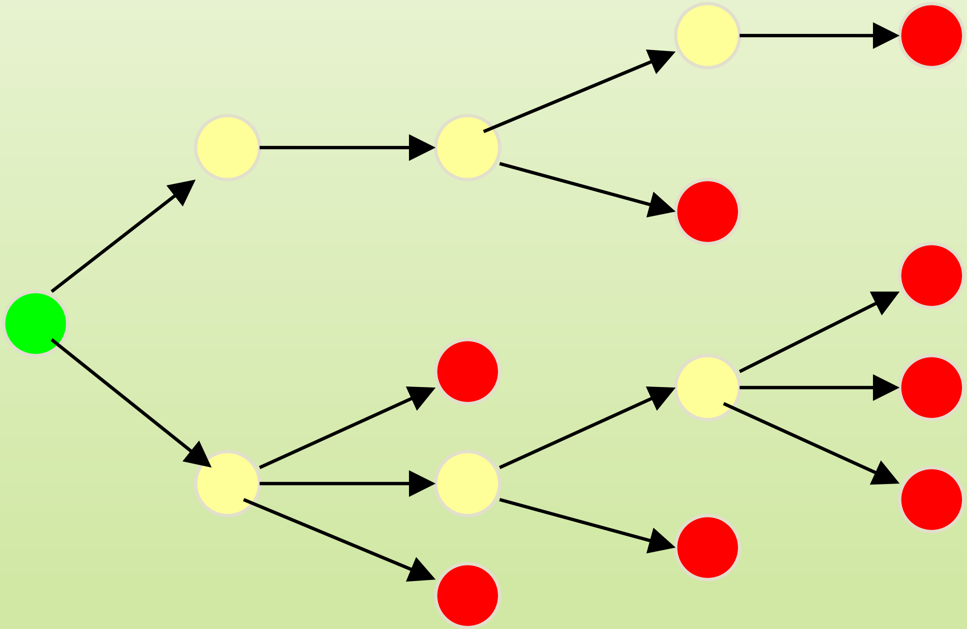
- **Ruang solusi diorganisasikan ke dalam struktur pohon.**
- **Tiap simpul pohon menyatakan status (*state*) persoalan, sedangkan sisi (cabang) dilabeli dengan nilai-nilai x_i.**
- **Lintasan dari akar ke daun menyatakan solusi yang mungkin.**
- **Seluruh lintasan dari akar ke daun membentuk ruang solusi.**
- **Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (*state space tree*).**

5. Pengorganisasian Solusi

- **Sebuah pohon adalah sekumpulan simpul dan busur yang tidak mempunyai sirkuit**
- **Ada tiga macam simpul:**



- ***Backtracking* dapat dipandang sebagai pencarian di dalam pohon menuju simpul daun (goal) tertentu**



*) Sumber: *www.cis.upenn.edu/.../35-**backtracking**.ppt*

5. Pengorganisasian Solusi

- **Tinjau persoalan *Knapsack* 1/0 untuk *n* = 3.**

$n = 3$.

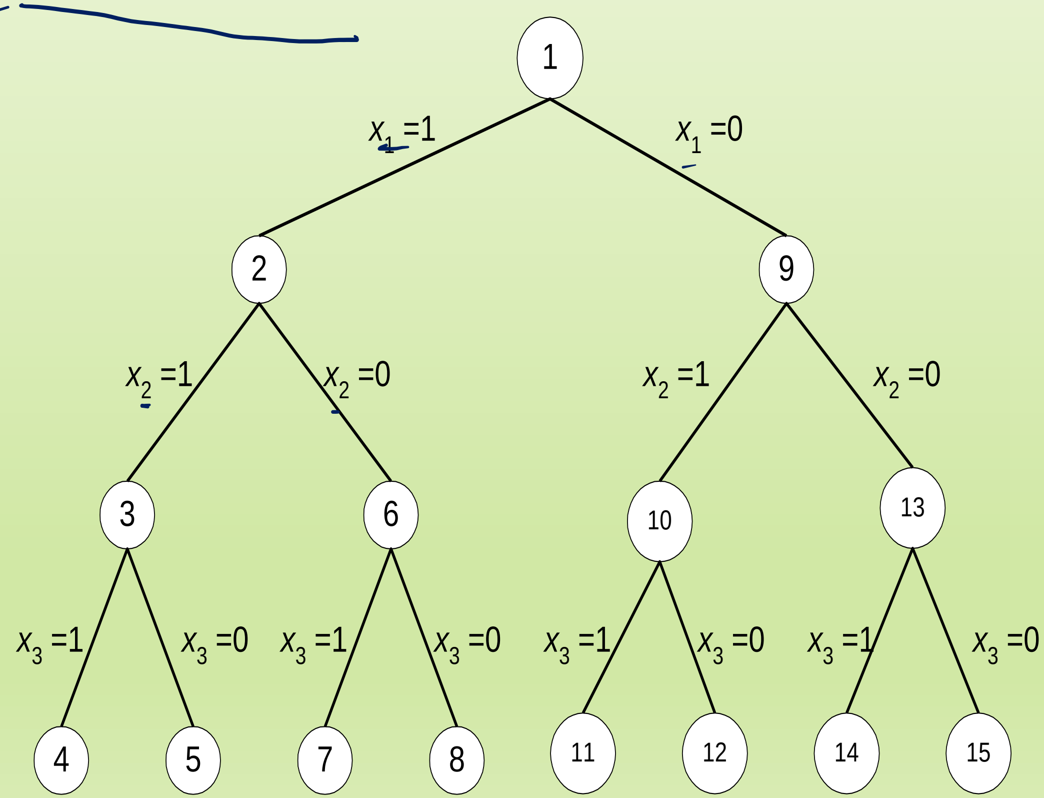
x_1, x_2, x_3 $x_i \in \{0, 1\}$.

- **Ruang solusinya:**

x_1

x_2

x_3



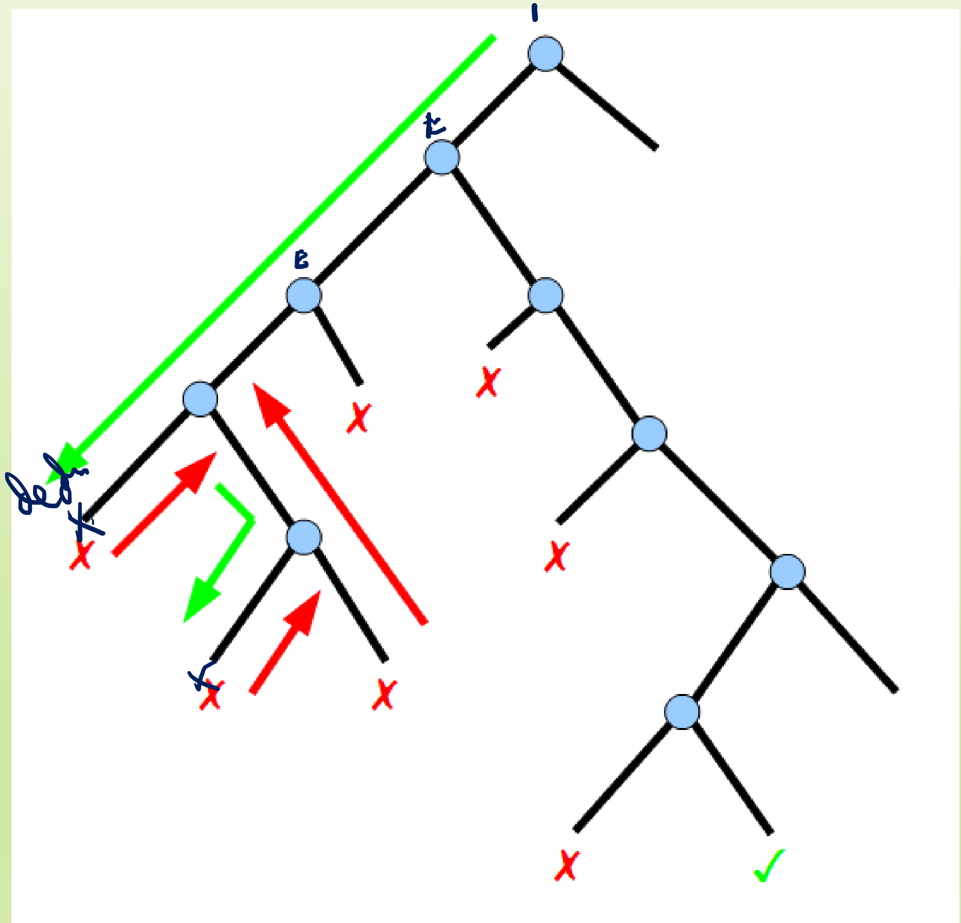
6. Prinsip Solusi dengan Metode Runut-balik

- **Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti aturan *depht-first order* (DFS).**
- **Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*).**
- **Simpul hidup yang *sedang* diperluas dinamakan simpul-E (*Expand-node*).**
- **Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang.**
- **Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" sehingga menjadi simpul mati (*dead node*).**
- **Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas (*bounding function*).**
- **Simpul yang sudah mati tidak akan pernah diperluas lagi.**
- **Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian *backtrack* ke simpul aras diatasnya**

FILKOM
FAKULTAS ILMU KOMPUTER

6. Prinsip Solusi dengan Metode Runut-balik

- **Lalu, teruskan dengan membangkitkan simpul anak yang lainnya.**
- **Selanjutnya simpul ini menjadi simpul-E yang baru.**
- **Pencarian dihentikan bila kita telah sampai pada *goal node*.**



Sumber: http://www.w3.org/2011/Talks/01-14-steven-phenotype/

Contoh 1. Knapsack 0/1

	1	2	3
	x_1	x_2	x_3
	1		
W	35	32	25

- **Tinjau persoalan *Knapsack* 0/1 dengan instansiasi:**

  ***n* = 3**

  x_1

  $(w_1, w_2, w_3) = (35, 32, 25)$

  $35 \leq 30$

  $(p_1, p_2, p_3) = (40, 25, 50)$

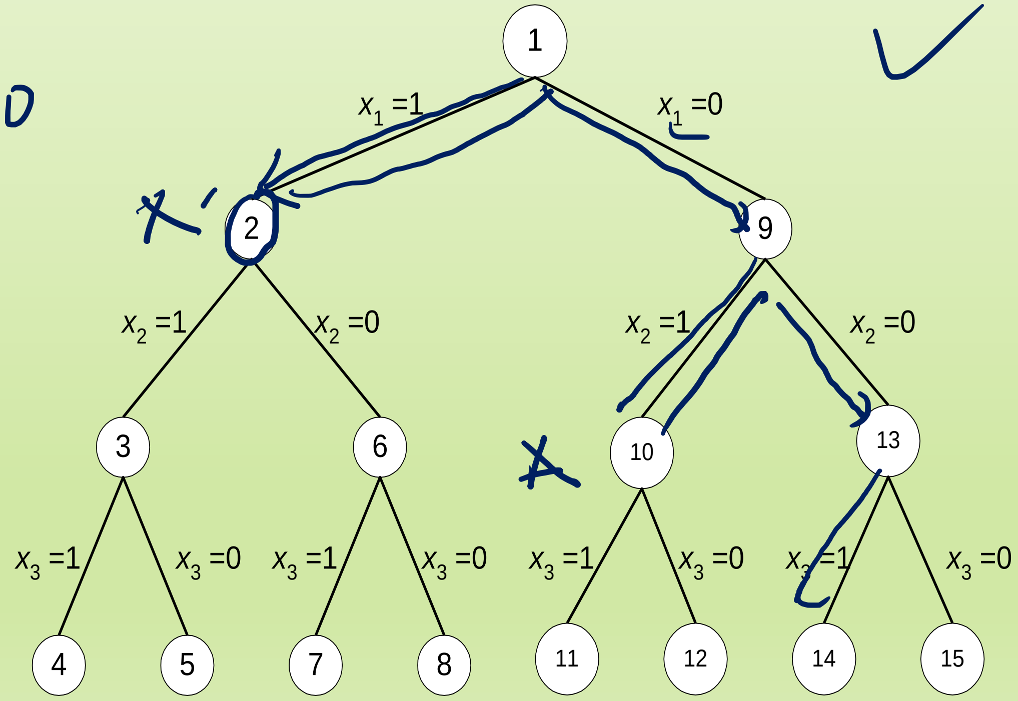
  $M = 30$

- **Solusi dinyatakan sebagai $X = (x_1, x_2, x_3)$, $x_i \in \{0, 1\}$.**
- **Fungsi konstrain (dapat dianggap sebagai *bounding function*):**

$$\sum_{i=1}^{k} w_i x_i \leq M$$

0 0 1

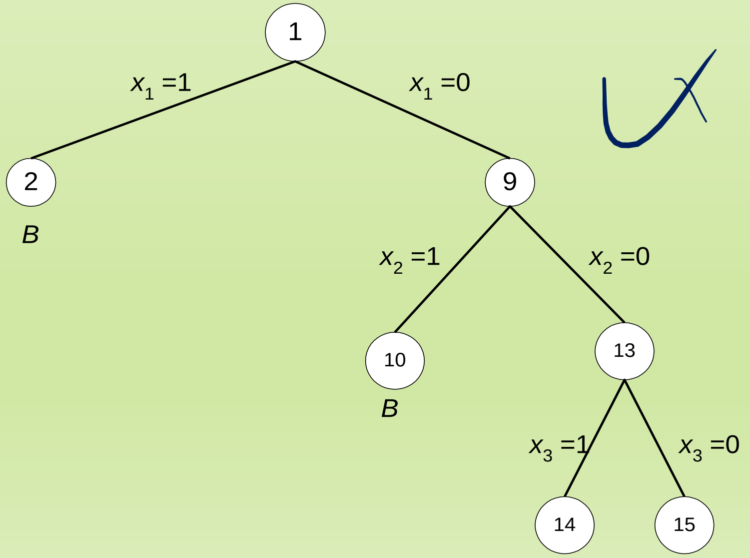
- **Pada metode *brute force*, semua lintasan dari akar ke daun dievaluasi**



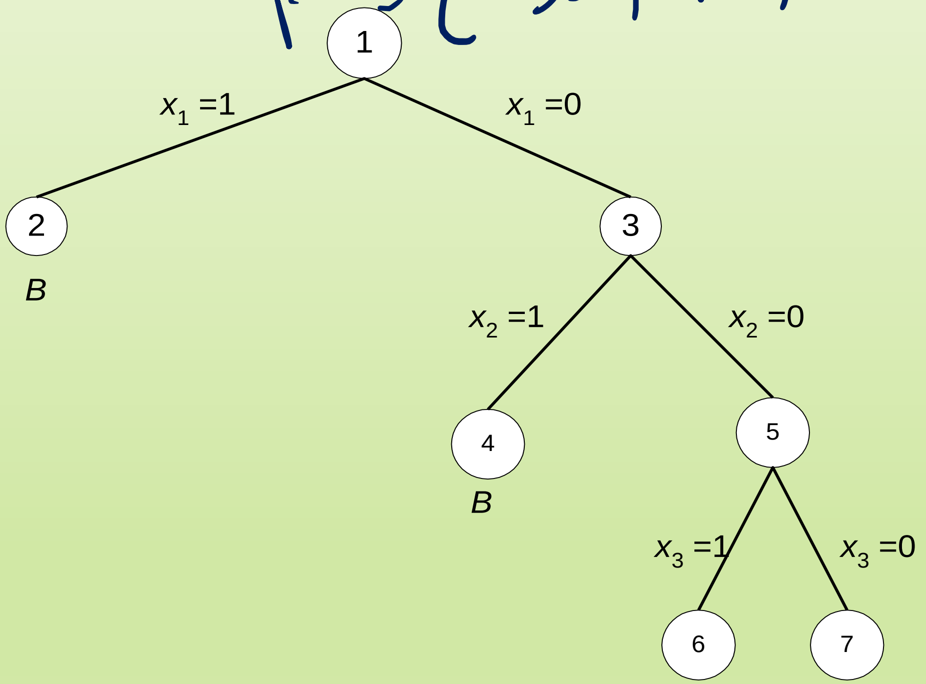
Contoh 1. Knapsack 0/1

$M = 35$
$w = (10, 20, 15, 25)$
$P = (35, 40, 50, 70)$

- **Pohon dinamis yang dibentuk selama pencarian untuk persoalan *Knapsack* 0/1 dengan $n = 3$, $M = 30$, $w = (35, 32, 25)$ dan $p = (40, 25, 50)$**



Penomeran Ulang



Solusi optimumnya adalah $X = (0, 0, 1)$ dan $F = 50$.

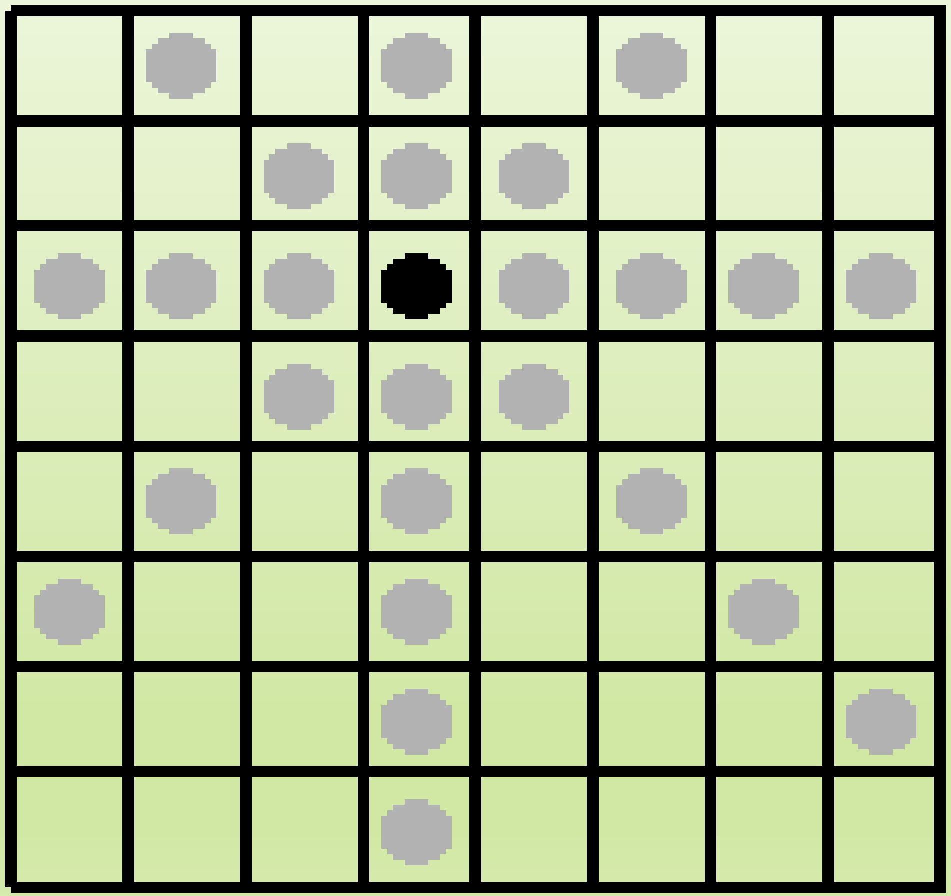
7. Skema Algoritma Runut-Balik (versi rekursif)

```
procedure RunutBalikR(input k:integer)
Algoritma:
  for tiap x[k] yang belum dicoba sedemikian sehingga
              ( x[k] ← T(k)) and B(x[1], x[2], ... ,x[k])= true do
    if (x[1], x[2], ... ,x[k]) adalah lintasan dari akar ke daun
    then
      CetakSolusi(x)
    endif
    RunutBalikR(k+1)   { tentukan nilai untuk x[k+1]}
  endfor
```

- **Setiap simpul dalam pohon ruang status berasosiasi dengan sebuah pemanggilan rekursif.**
- **Jika jumlah simpul dalam pohon ruang status adalah 2^n atau $n!$, maka untuk kasus terburuk, algoritma runut-balik membutuhkan waktu dalam $O(p(n)2^n)$ atau $O(q(n)n!)$,**
- **dengan $p(n)$ dan $q(n)$ adalah polinom derajat n yang menyatakan waktu komputasi setiap simpul.**

Contoh 2. Persoalan N-Ratu

- **Diberikan sebuah papan catur yang berukuran *N* × *N* dan delapan buah ratu. Bagaimanakah menempatkan *N* buah ratu (Q) itu pada petak-petak papan catur sedemikian sehingga tidak ada dua ratu atau lebih yang terletak pada satu baris yang sama, atau pada satu kolom yang sama, atau pada satu diagonal yang sama?**



Contoh 2 buah solusi *8-queen problem*:

5x5 ·

$x_7 = 1$

$x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8$

1 3 5 8

X	Q						
			Q				
					Q		
							Q
		Q					
Q							
						Q	
				Q			

							Q
		Q					
Q							
					Q		
	Q						
				Q			
						Q	
			Q				

Contoh 2 buah solusi *8-queen problem : BF*

a) Brute Force 1

- Mencoba semua kemungkinan solusi penempatan delapan buah ratu pada petak-petak papan catur.
- Ada $C(64, 8) = 4.426.165.368$ kemungkinan solusi.

b) Brute Force 2

- Meletakkan masing-masing ratu hanya pada baris-baris yang berbeda. Untuk setiap baris, kita coba tempatkan ratu mulai dari kolom 1, 2, …, 8.
- Jumlah kemungkinan solusi yang diperiksa berkurang menjadi

$$8^8 = 16.777.216$$

c) Brute Force 3 (exhaustive search)

- Misalkan solusinya dinyatakan dalam vektor 8-*tupple*:

$$X = (x_1 , x_2 , \dots , x_8)$$

- Vektor solusi merupakan permutasi dari bilangan 1 sampai 8.
- Jumlah permutasi bilangan 1 sampai 8 adalah $P(1, 8) = 8! = 40.320$ buah.

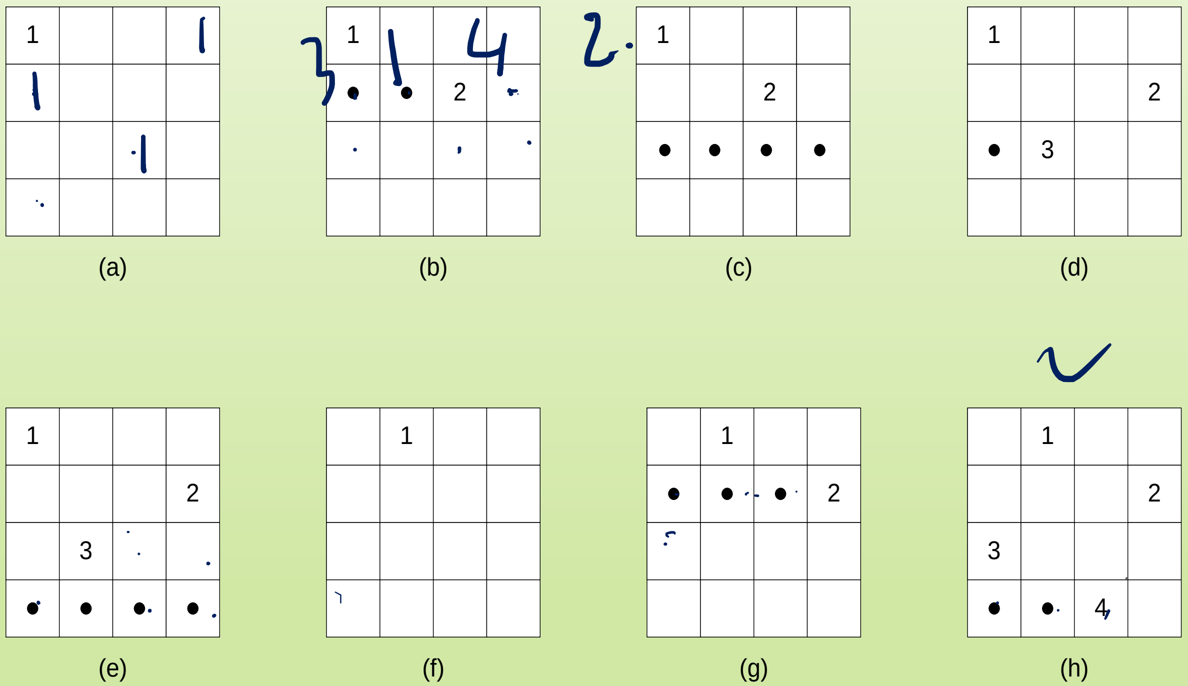
Contoh 2 buah solusi *8-queen problem* : RB

- **Algoritma runut-balik memperbaiki algoritma *brute force* 3 (*exhaustive search*).**
- **Ruang solusinya adalah semua permutasi dari angka-angka 1, 2, 3, 4, 5, 6, 7, 8.**
- **Setiap permutasi dari 1, 2, 3, 4, 5, 6, 7, 8 dinyatakan dengan lintasan dari akar daun. Sisi-sisi pada pohon diberi label nilai *xi*.**

$x_1\ x_2\ x_3\ x_4$

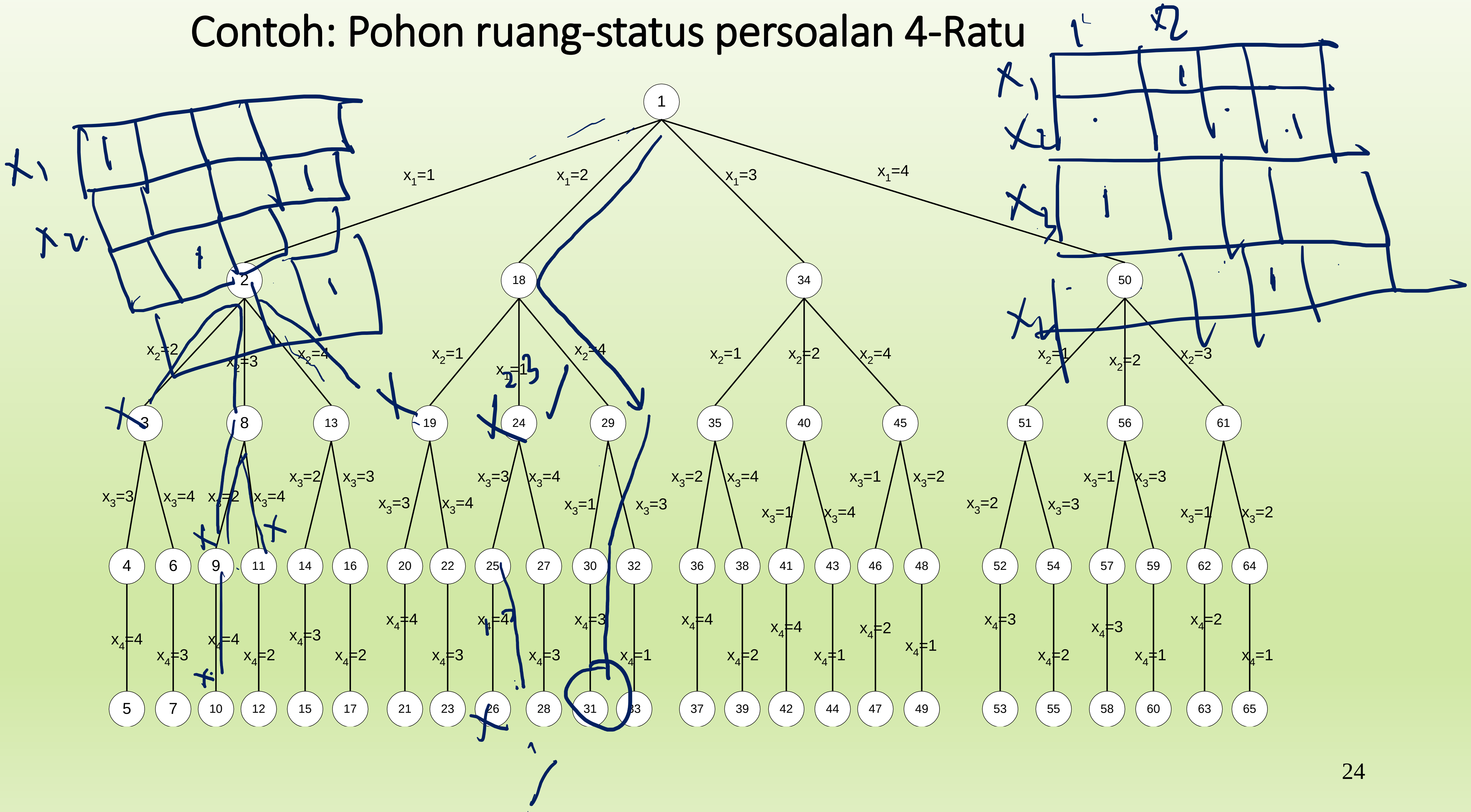
2 4 1 3

3 1 4 2

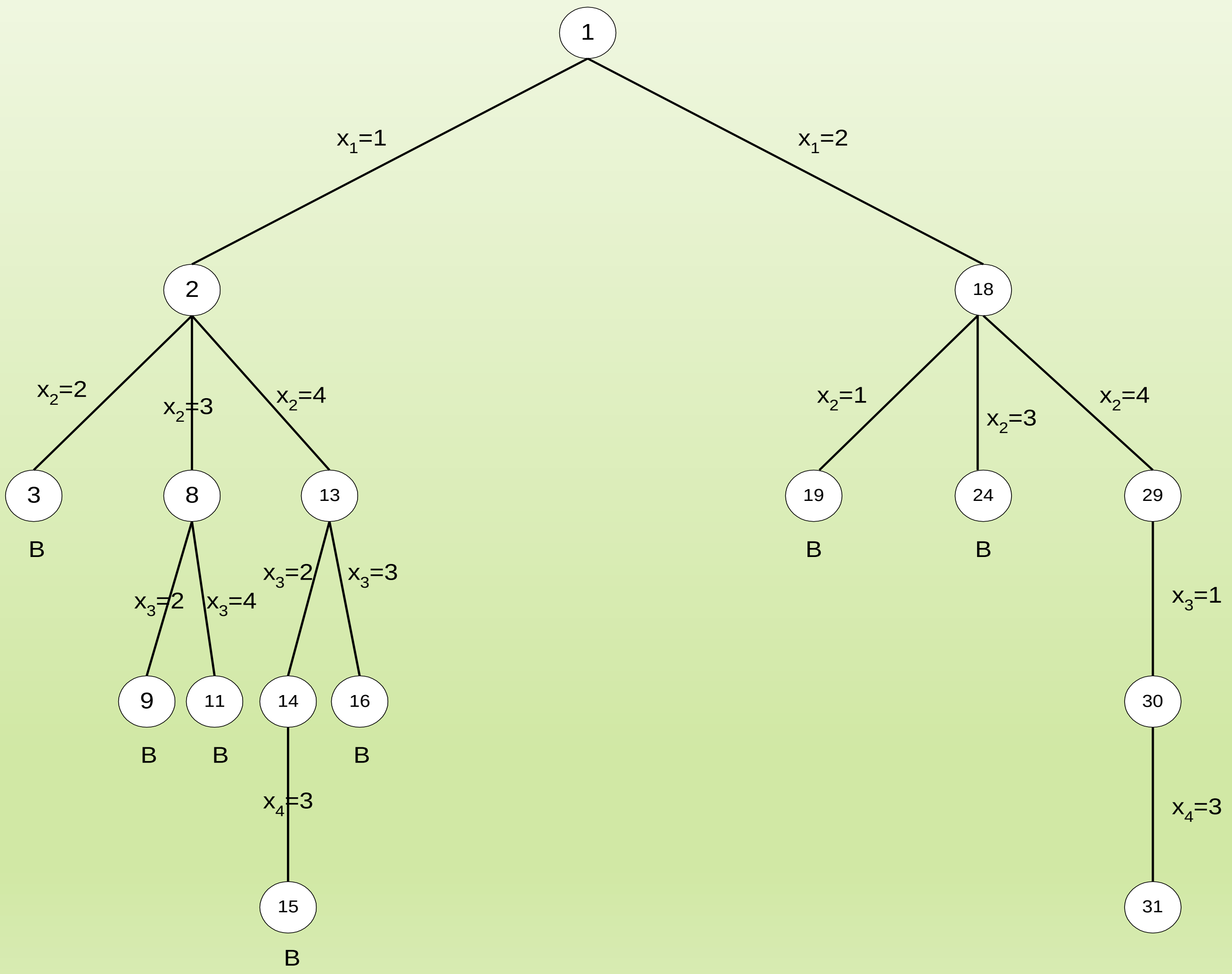


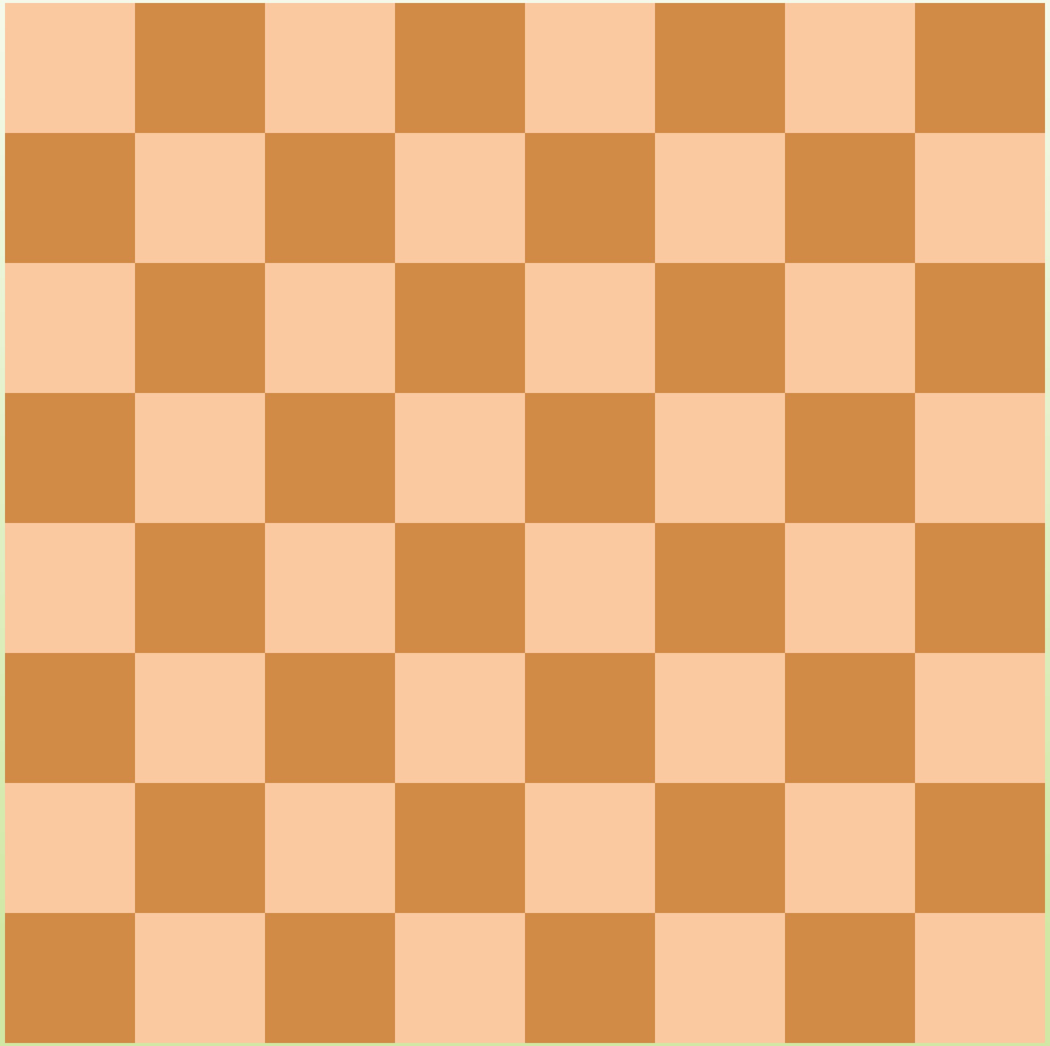
FILKOM
FAKULTAS ILMU KOMPUTER

Contoh: Pohon ruang-status persoalan 4-Ratu



Pohon ruang status dinamis persoalan 4-Ratu yang dibentuk selama pencarian:





Algoritma Runut-balik untuk Persoalan 8-Ratu

- **Tinjau dua posisi ratu pada (i, j) dan (k, l)**
- **Dua buah ratu terletak pada baris yang sama, berarti**

$$i = k$$

- **Dua buah ratu terletak pada kolom yang sama, berarti**

$$j = l$$

- **Dua buah ratu terletak pada diagonal yang sama, berarti**

$$\searrow\ i - j = k - l \ \text{ atau } \ \swarrow\ i + j = k + l$$

$$\Leftrightarrow i - k = j - l \ \text{ atau } \ k - i = j - l$$

$$\Leftrightarrow |j - l| = |i - k|$$

Algoritma Runut-balik untuk Persoalan 8-Ratu

Algoritma:

- **Inisialisasi *x*[1], *x*[2], …, *x*[*N*] dengan 0**

  for i←N to n do

  x[i]←0

  endfor

- **Panggil prosedur N_RATU_R(1)**

```
procedure RunutBalikR(input k:integer)
Algoritma:
  for tiap x[k] yang belum dicoba sedemikian sehingga
              ( x[k] ← T(k)) and B(x[1], x[2], ... ,x[k])= true do
    if (x[1], x[2], ... ,x[k]) adalah lintasan dari akar ke daun
    then
       CetakSolusi(x)
    endif
    RunutBalikR(k+1)   { tentukan nilai untuk x[k+1]}
  endfor
```

Algoritma Runut-balik untuk Persoalan 8-Ratu

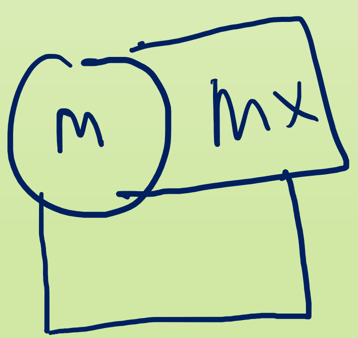
```
procedure N_RATU_I(input N:integer)
Deklarasi
   k : integer
Algoritma:
   k ← 1        {mulai pada baris catur ke-1}
   x[1] ← 0   {inisialisasi kolom dengan 0}
   while k > 0 do
        x[k] ← x[k]+1 {pindahkan ratu ke kolom berikutnya}
        while (x[k] > N) and (not TEMPAT(k)) do
                x[k] ← x[k] + 1
        endwhile
        if x[k] < n then
                if k=N then   { solusi sudah lengkap?}
                        CetakSolusi(x,N)     { cetak solosi}
                else
                        k ← k+1{pergi ke baris berikutnya}
                        x[k] ← 0{inisialisasi kolom dengan 0}
                endif
        else
                k ← k-1{runut-balik ke baris sebelumnya}
        endif
 endwhile
```

```
function TEMPAT(input k:integer) ← boolean
{true jika ratu dapat ditempatkan pada kolom x[k], false jika tidak}
Deklarasi
  i : integer
  stop : boolean
Algoritma:
  kedudukan⍰true   { asumsikan ratu dapat ditempatkan pada kolom x[k] }
  { periksa apakah memang ratu dapat ditempatkan pada kolom x[k] }
  i ← 1              { mulai dari baris pertama}
  stop ← false
  while (il < k ) and (not stop) do
    if (x[i]=x[k]){apakah ada dua buah ratu pada kolom yang sama?}  or
     (ABS(x[i]-x[k])=ABS(i-k)) {dua ratu pada diagonal yang sama?}   then
     Kedudukan ← false
     Keluar ← ⍰true
    else
     i ← i+1           { periksa pada baris berikutnya}
    endif
  endwhile
  return kedudukan
```

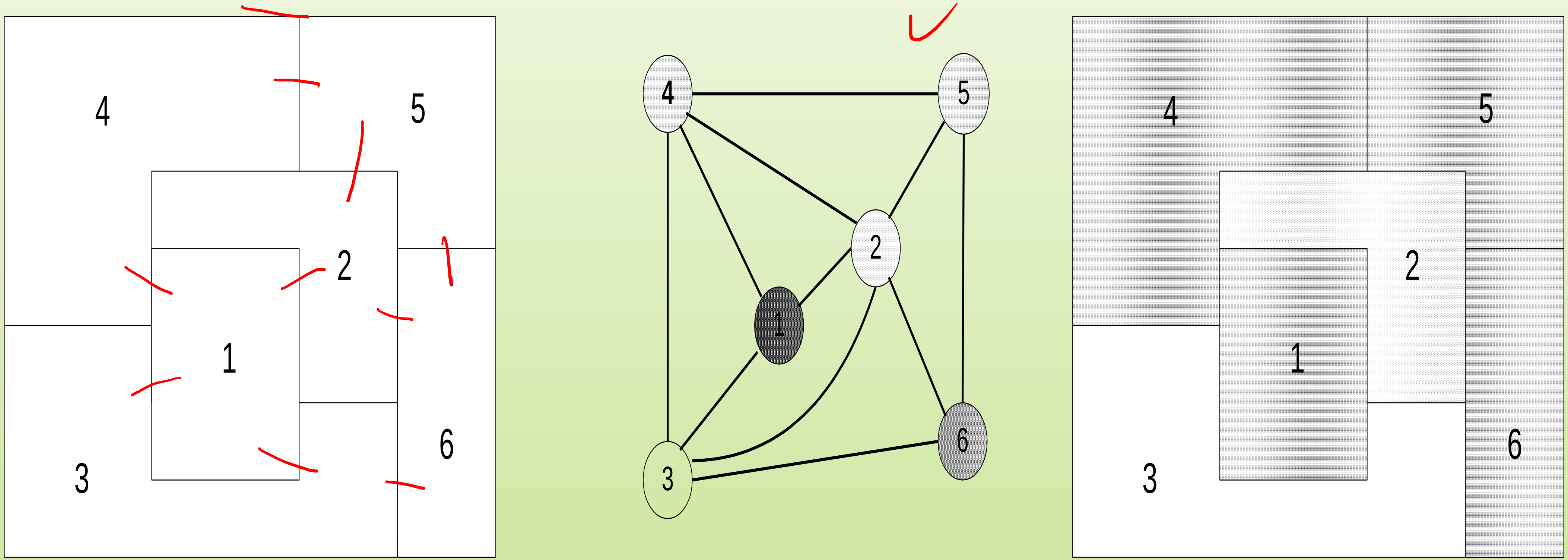
8. Pewarnaan Graf (*Graph Colouring)*

Persoalan:

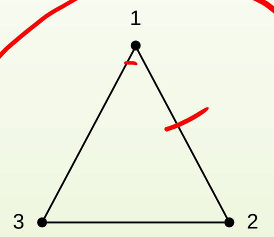
- **Diberikan sebuah graf *G* dengan *n* buah simpul dan disediakan *m* buah warna. Bagaimana mewarnai seluruh simpul graf *G* sedemikian sehingga tidak ada dua buah simpul bertetangga yang mempunyai warna sama (Perhatikan juga bahwa tidak seluruh warna harus dipakai)**



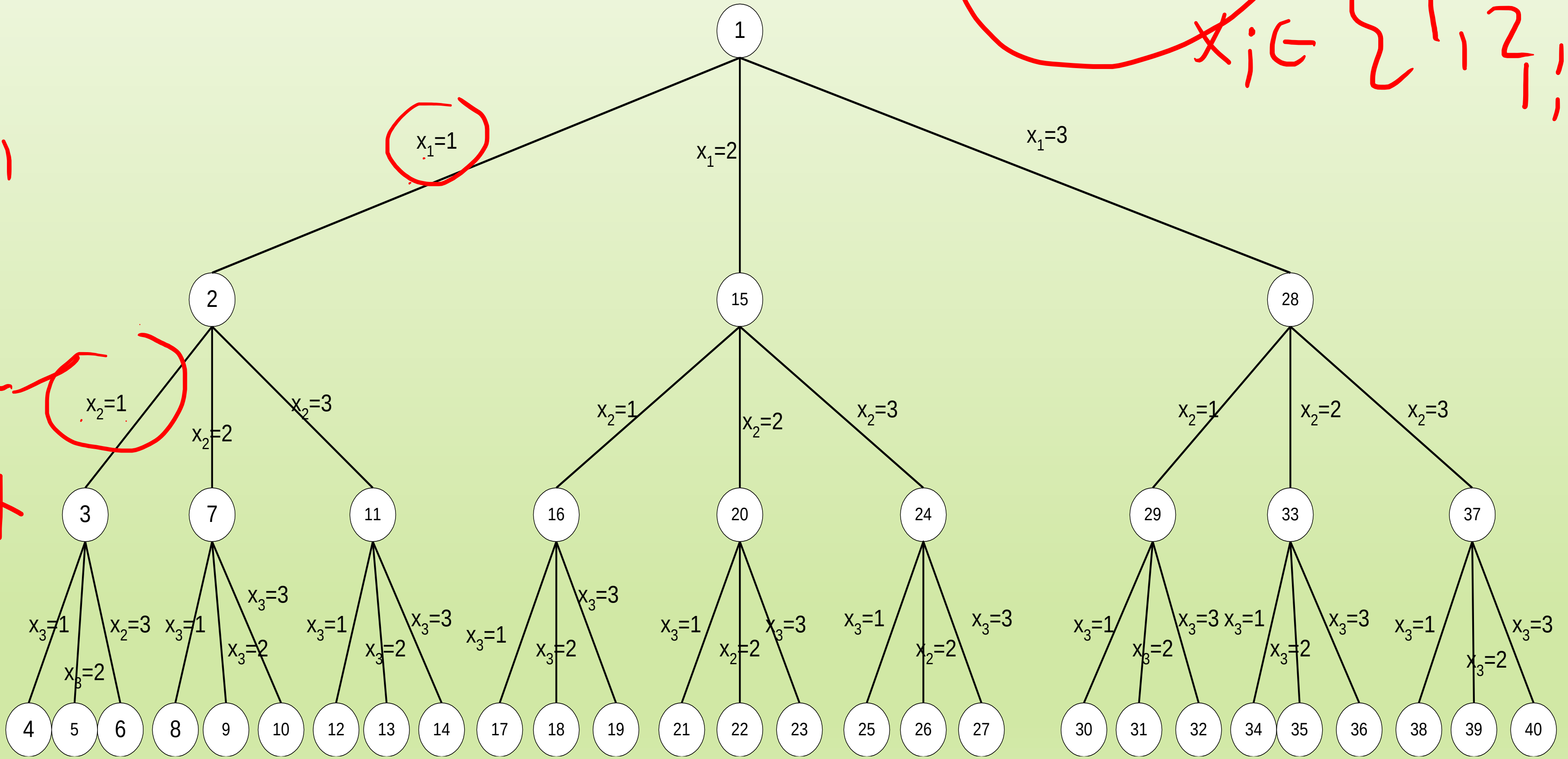
Contoh aplikasi: pewarnaan peta



Tinjau untuk n = 3 dan m = 3.

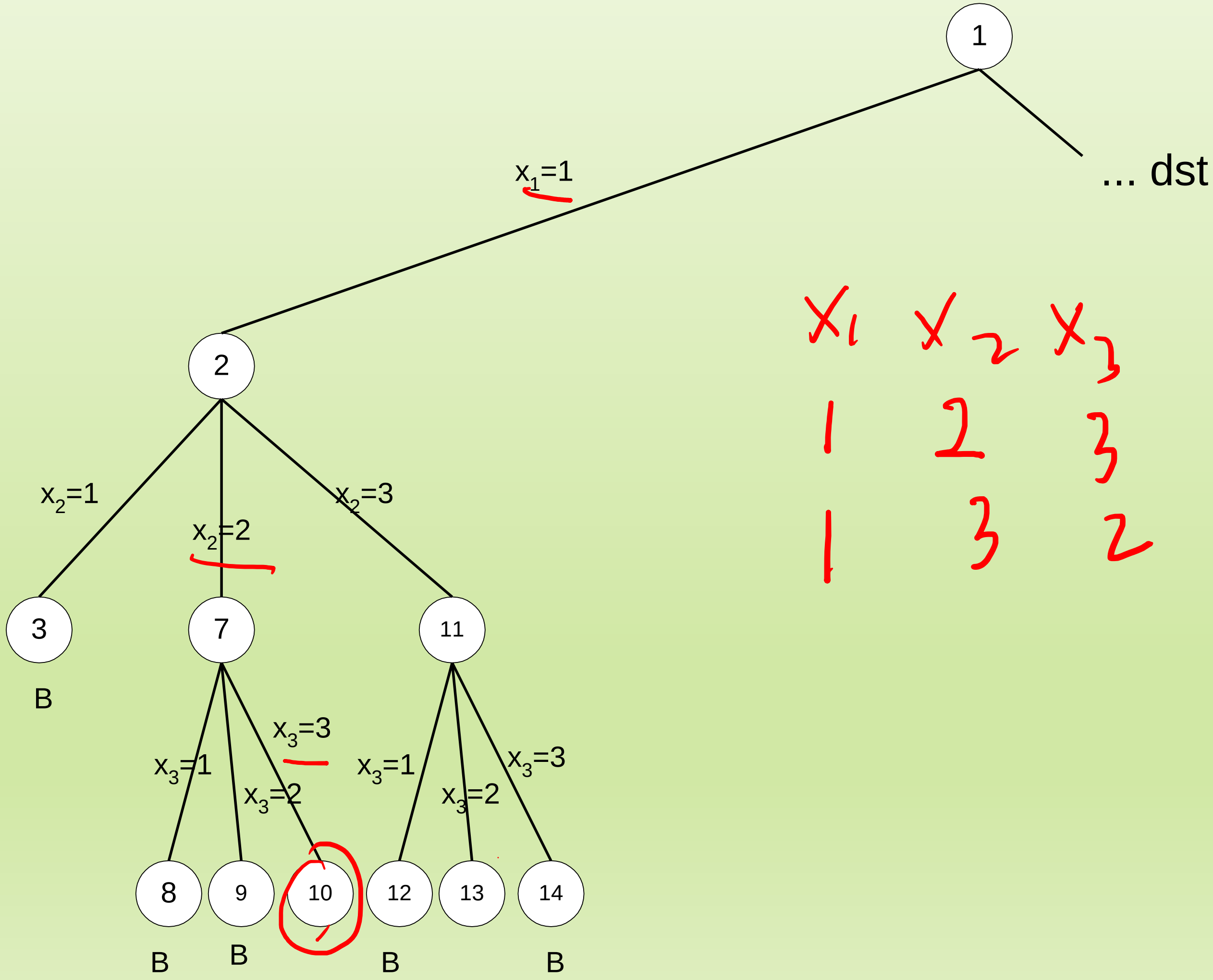


$x_i \in \{1, 2, 3\}$



Misalkan warna dinyatakan dengan angka 1, 2, …, *m* dan solusi dinyatakan sebagai vektor X dengan *n-tuple*:

$X = (x_1, x_2, \ldots, x_n)$, $x_i \in \{1, 2, \ldots, m\}$



x_1	x_2	x_3
1	2	3
1	3	2

Algoritma Runut-balik Untuk Pewarnaan Graf

- Masukan:

  1. Matriks ketetanggan GRAF[1..n, 1..n]

     GRAF[i,j] = true jika ada sisi (i,j)

     GRAF[i,j] = false jika tidak ada sisi (i,j)

  2. Warna

     Dinyatakan dengan integer 1, 2, ...,m

- Keluaran:

  1. Tabel X[1..n], yang dalam hal ini, x[i] adalah warna untuk simpul i.

- Algoritma:

  1. Inisialisasi x[1..n] dengan 0

     for i$\leftarrow$1 **to** n **do**

     x[i]$\leftarrow$0

     endfor

  2. Panggil prosedur PewarnaanGraf(1)

```
procedure PewarnaanGraf(input k : integer)
Deklarasi
  stop : boolean
Algoritma:
  stop ← false
  while not stop do
     {tentukan semua nilai untuk x[k] }
     WarnaBerikutnya(k) {isi x[k] dengan sebuah warna}
     if x[k] = 0 then    {tidak ada warna lagi, habis}
    stop ← true
     else
    if k = n then               {apakah seluruh simpul sudah diwarnai?}
       CetakSolusi(X,n)
    else
       PewarnaanGraf(k+1) {warnai simpul berikutnya}
      endif
    endif
  endwhile
```

```
procedure WarnaBerikutnya(input k:integer)
Deklarasi
  stop, keluar : boolean
  j : integer
Algoritma:
  stop ← false
  while not stop do
    x[k] ← (x[k]+1) mod (m+1)    {warna berikutnya}
    if x[k] = 0  then            {semua warna telah terpakai}
        stop ← true
    else {periksa warna simpul-simpul tetangganya}
        J ← 1
        Keluar← false
        while (j  ≤ n) and (not keluar) do
               if (GRAF[k,j]) {jika ada sisi dari simpul k ke simpul j} and
                     (x[k] = x[j]){warna simpul k = warna simpul j }  then
                            Keluar ← true {keluar dari kalang}
                     else
                            j ← j+1{periksa simpul berikutnya}
                     endif
        endwhile
        if j = n+1 {seluruh simpul tetangga telah diperiksa dan
                ternyata warnanya berbeda dengan x[k] } then
                     stop ← true {x[k] sudah benar, keluar dari kalang}
        endif
    endif
  endwhile
```

Kompleksitas Waktu algoritma PewarnaanGraf

- **Pohon ruang status yang untuk persoalan pewarnaan graf dengan n simpul dan m warna adalah pohon m-ary dengan tinggi n + 1.**
- **Tiap simpul pada aras i mempunyai m anak, yang bersesuaian dengan m kemungkinan pengisian x[i], untuk 1 ≤ i ≤ n.**
- **Simpul pada aras n adalah simpul daun. Jumlah simpul internal (simpul bukan daun) ialah**
- **Tiap simpul internal menyatakan pemanggilan prosedur WarnaBerikutnya yang membutuhkan waktu dalam O(mn).**
- **Total kebutuhan waktu algoritma PewarnaanGraf adalah**

$$\sum_{i=1}^{n} m^i n = \frac{n(m^{n+1}-1)}{(m-1)} = O(nm^n)$$

Latihan

- **(*Sum of subset problem*) Diberikan sebuah himpunan *A* yang berisi *n* buah bilangan positif berbeda dan sebuah bilangan bulat *m*. Tentukan himpunan bagian dari himpunan bilangn bulat tersebut yang jumlahnya sama dengan *m*.**

Contoh: A = {5, 6, 10, 16} dan *m* = 21.

Himpunan bagian yang memenuhi:

{5, 6, 10}, {5, 16}

Selesaikan dengan algoritma runut-balik!